Community Detection in Networks: The Leader-Follower Algorithm

Devavrat Shah and Tauhid Zaman*
devavrat@mit.edu, zlisto@mit.edu

November 4, 2010

Abstract

Traditional spectral clustering methods cannot naturally learn the number of communities in a network and often fail to detect smaller community structure in dense networks because they are based upon *external* community connectivity properties such as graph cuts. We propose an algorithm for detecting community structure in networks called the leader-follower algorithm which is based upon the natural *internal* structure expected of communities in social networks. The algorithm uses the notion of network centrality in a novel manner to differentiate *leaders* (nodes which connect different communities) from *loyal followers* (nodes which only have neighbors within a single community). Using this approach, it is able to naturally learn the communities from the network structure and does not require the number of communities as an input, in contrast to other common methods such as spectral clustering. We prove that it will detect all of the communities *exactly* for *any* network possessing communities with the natural internal structure expected in social networks. More importantly, we demonstrate the effectiveness of the leader-follower algorithm in the context of various real networks ranging from social networks such as Facebook to biological networks such as an fMRI based human brain network. We find that the leader-follower algorithm finds the relevant community structure in these networks without knowing the number of communities beforehand. Also, because the leader-follower algorithm detects communities using their internal structure, we find that it can resolve a finer community structure in dense networks than common spectral clustering methods based on external community structure.

1 Introduction

Social networks often possess a structure which reflects the underlying communities which make up a population. These networks are very often unweighted, have a high edge density, and an unknown number of communities. This makes it challenging to detect the finer community structure with existing methods. Typically, only large scale, coarse community structure can be detected, while smaller communities go unseen. By not detecting this finer structure, we may lose valuable information about the network and the social interactions it represents. What we would like is a method which can find communities of any size in a network without requiring one to know the number of communities beforehand.

1.1 Prior work

Finding communities is essentially a data clustering problem, where the goal is to assign each node to a community or cluster in some reasonable manner. There has been much prior work on clustering and community detection in networks. Here we shall be able to describe some of the relevant prior work. The most common method for community detection is spectral clustering which is based on optimizing graph-cuts [1],[2]. There are also other approaches which which use a statistical inference approach [3],[4],[5],[6] and hierarchical methods [7]. An interested reader is referred to detailed surveys [8] and [9]. Here we will focus on spectral clustering as it is one the most common and widely used clustering techniques.

There are two types of graph cuts used in spectral clustering: RatioCut [1] and NCut [2]. These functions are minimized when the nodes are grouped into large communities with few inter-community edges. Minimizing graph-cuts is in general NP-hard, but an approximation can be found using spectral clustering. Here one finds the eigenvectors corresponding to the smallest k eigenvalues of the Laplacian matrix of the network and arranges these column vectors into an $n \times k$ matrix. One then treats the rows of this matrix as data points for each node in the network and clusters them using techniques such as k-means clustering.

Spectral clustering requires one to specify the number of communities in the network. This is a major weakness of this method in situations where this information is not known beforehand. Also, as we will show in this work, spectral clustering is unable to resolve smaller communities in densely connected networks.

1.2 Our contributions

There are two key contributions of this work. First, using insights from social networks, we propose a certain natural internal structure for a community. Second, we provide an algorithm that uses this internal structure to find these communities. We prove that this algorithm, called *leader-follower*, is guaranteed to find communities with the proposed structure in any network in an efficient manner. It is non-parametric, does require one to know the number of communities beforehand, and has no resolution limit with regard to community size. We demonstrate the algorithm's effectiveness by means of various experiments on synthetic and real networks.

We start by using intuition drawn from social networks to propose two structural properties one would expect a community to possess. First, everyone in a community should know each other, or more formally, the community should be a *clique*. Second, each community should possess distinguishing members. More formally, each community must have at least one *loyal follower*, which is a node in the community with no neighbors in any other community. Loyal followers are what provide the community with a distinguishable identity.

These properties are essentially *internal* to the community, and are invariant to the density of inter-community edges. We show that the performance of spectral clustering degrades as the inter-community edge density increases. This is because spectral clustering is based on *external* community properties (graph-cut) which are dependent upon the inter-community edge density.

To detect communities in a manner which is robust to the inter-community edge density, we propose the leader-follower algorithm, which is based upon internal community structure. The algorithm has roughly two stages. First, it detects "leaders" in the network, where a node is a "leader" if it is *not* a loyal follower for any community. This differentiation of leaders and loyal followers is accomplished through the novel use of a certain notion of network centrality. Second, the algorithm assigns the remaining "loyal followers" to the leaders in a specific manner in order to form the communities.

The leader-follower algorithm is able to naturally learn the number and size of the communities using the network's structure. In addition, we are able to prove that for any network where the communities possess the proposed structure, the leader-follower algorithm will detect all of the communities exactly, independent of the inter-community edge density.

To establish the performance improvement provided by the use of internal rather than external community structure, we conduct a variety of experiments on synthetic and real networks, comparing the leader-follower algorithm with the more common spectral clustering method. These networks range from social networks such as Facebook to biological networks such as the human brain. We find that in each network, the leader-follower algorithm can resolve a finer community structure than spectral clustering without needing to know the number of communities beforehand.

2 Community structure

2.1 Community structure

When thinking about what defines a community in a social network, two natural properties come to mind. First, people in a community should know each other, or the community should have a high edge density. Second, each detectable community ought to have a unique identity. If everyone in a community also belonged to other communities, it would be difficult to distinguish the original community. Therefore, each community needs at least one *loyal* member or follower who is not a member of any other community. This loyal follower can be thought of as the detectable *signal* of a community. We formalize these properties in the following definition.

Definition 1 (**Community Structure**). *Given a network graph $G(V, E)$ with node set V and edge set E, a set of nodes $C \subseteq V$ is called a community if it satisfies the following properties:*

Property 1: Strong Connectedness. *The induced subgraph of the set C is a clique.*

Property 2: Loyal Follower. *There exists at least one node $f \in C$ such that f has no neighbors outside of C.*

We say that a network $G(V, E)$ has a community structure satisfying Definition 1 if there exist disjoint subsets $C_1, \ldots, C_k$ of V for some $k \geq 1$ such that each $C_i,\ 1 \leq i \leq k$ satisfies the "Strong Connectedness" and "Loyal Follower" properties.

2.2 Internal vs. external community properties

The above community properties are essentially *internal* and do not depend on the connections with the rest of the network. In contrast, spectral methods based on optimizing graph cuts use the external connectivity (the cut value) to identify communities. When this external connectivity is too dense, it is expected that spectral methods will not be able to correctly identify smaller scale community structure. To demonstrate this, consider the two networks shown in Figure 1. These networks each satisfy Definition 1 (communities are cliques and have at least one loyal follower). They each contain 104 nodes grouped into the same 10 communities with sizes chosen uniformly between 2 and 30. One network was sparse and contained 200 inter-community edges and the other was dense and contained 2000 inter-community edges. The resultant communities found by spectral clustering are shown in Figure 1. The true communities are indicated by color and grouped together. The erroneous clusters found with spectral clustering (with the true number of communities as input) are enclosed with white lines in the figure. Spectral clustering ends up combining smaller communities into larger communities in the denser network and leaves some single node communities.

As a quantitative measure, we define an error function for the clustering as follows. For a network $G(V, E)$ with N nodes and a set of k communities found by a community detection algorithm $C_1, C_2, ..., C_k$, let M_{ij} be 1 if nodes i and j are in the same true community and put in different communities by the algorithm or if they are in different true communities and put in the same community by the algorithm, and 0 otherwise. Then the error function is

$$E(C_1, C_2, ..., C_k) = \sum_{i=1}^{N} \sum_{j=i}^{N} M_{ij} \tag{1}$$

This function counts the number of distinct misclassified node pairs. The bar graph in Figure 1 shows the error of spectral clustering for the sparse and dense networks. As can be seen, the error increases as the graph becomes more dense. Detection using external community properties is no longer able to see the smaller communities because of these 'noisy' inter-community edges. However, we will see that if we use internal properties, we can detect fine community structure even in dense networks.

2.3 Loyal followers and distance centrality

In order to use internal properties to find communities, we must have a way to detect these internal properties. One of the properties was the notion of a loyal follower which gave the community its identity. If we could detect these loyal followers, we would be able to find the communities. One way to find loyal followers is to assign each node a score function which can differentiate loyal followers from the other nodes, which we will refer to as *leaders*. A good score function is the graph theoretic notion of **distance centrality** [10]. Distance centrality measures how close a node is to all other nodes in the network. Let $d(u, v)$ denote the shortest path distance between nodes u and v in a

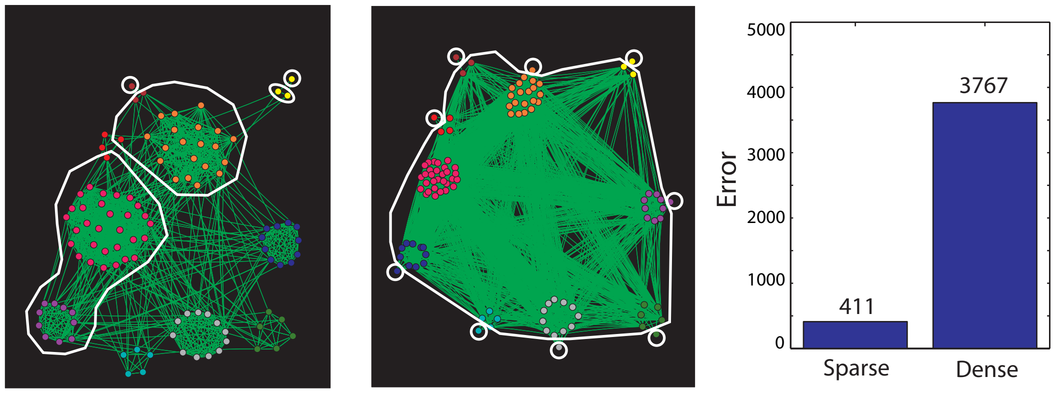

Figure 1: Communities found in synthetic networks with (left) sparse and (middle) dense inter-community connectivity. The true communities are indicated by color and grouped together. The clusters found with spectral clustering which are not the true communities are enclosed with white lines. (right) Bar chart showing the error of spectral clustering for the two networks.

network G. Then the distance centrality $D(u)$ of $u \in V$ is

$$D(u) = \sum_{v \in G} d(u, v). \tag{2}$$

Loyal followers should have higher distance centralities than leaders because they must pass through a leader to reach any node outside of their community. The problem is that the distance centrality of a loyal follower is dependent upon the size of its community. Therefore, on its own, distance centrality will not be able to single out loyal followers. This can be seen in Figure 2, where we plot the distance centrality for the nodes in the network shown. Different levels of distance centrality are seen which correspond to the communities of different size, but no clear loyal follower signal is evident. However, we note that within a community, the distance centrality of a loyal follower is always less than that of a leader. This can be seen in the plot on the right of Figure 2, where we have grouped the nodes by their community membership. In the figure, one can see local spikes of distance centrality corresponding to the loyal followers. Thus, if the distance centrality of a node is greater than or equal to *all* of its neighbors, it will be a loyal follower. This differential measurement using distance centrality is how we will detect loyal followers.

3 Leader-follower algorithm

Using the ideas developed thus far, we now present the leader-follower algorithm for detecting community structure in networks. Roughly speaking, there are two stages in the algorithm. First, we distinguish leaders from loyal followers in the network and

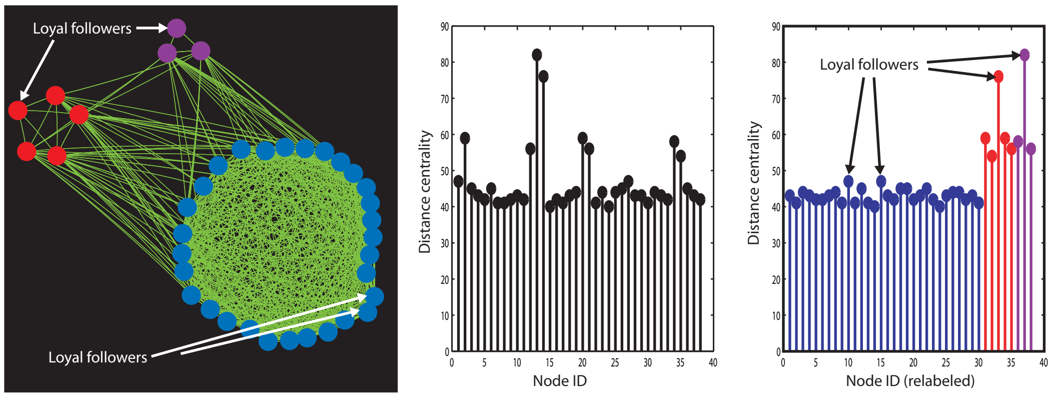

Figure 2: (left) An example of a network satisfying Definition 1. The communities are indicated by color and grouping and the loyal followers of each community are labeled. (middle) Plot of the distance centrality of the nodes in the network. (right) Plot of the distance centrality of the nodes in the network, with the nodes grouped by their community.

second, we assign loyal followers to leaders in order to form the communities. We now describe these steps in more detail.

3.1 Leader and loyal follower detection

We use the ideas from Section 2.3 to detect leaders and loyal followers. We first calculate the distance centrality of each node in the network. Then, leaders will be any node whose distance centrality is less than at least one of its neighbors, and the remaining nodes will be loyal followers. We prove in Section 5 that this procedure will correctly identify leaders and loyal followers in any network satisfying the properties of Definition 1.

Summary of leader and loyal follower detection step

1. For each $v \in V$, calculate its distance centrality $D(v)$.
2. Define leader nodes as

$$\mathcal{L} = \{v \in V : \text{there exists } u \in V \text{ such that } (u, v) \in E \text{ and } D(v) < D(u)\}.$$

and loyal follower nodes as $\mathcal{F} = V \backslash \mathcal{L}$

3.2 Community assignment

Once the leader and loyal follower sets have been detected, community assignment is the next step. We define a membership function $M : V \to V \cup \{\star\}$ that maps each $v \in V$ to a 'leader node', say $M(v) \in V$, who will identify a community to which v belongs: $C_v = \{u \in V : M(u) = v\}$. To start with, for each leader $v \in \mathcal{L}$, set $M(v) = v$; for each loyal follower $v \in \mathcal{F}{=}V\backslash\mathcal{L}$ set $M(v) = \star$. Sort the leaders in order of increasing distance centrality: let $\mathcal{L} = \{v_1, v_2, ..., v_{|\mathcal{L}|}\}$ with $D(v_i) \leq D(v_{i+1})$ for $1 \leq i < |\mathcal{L}|$. Now for each $u \in \mathcal{F}$ such that $(v_1, u) \in E$, set $M(u) = v_1$. After assigning loyal followers that are neighbors of $v_1, \ldots, v_i$, consider leader v_{i+1}. If $u \in \mathcal{F}$ such that $(u, v_{i+1}) \in E$ and $M(u) = \star$, then assign $M(u) = v_{i+1}$. The purpose of this stage is to seed the communities using loyal followers.

At the end of the first assignment step we may have additional spurious communities, which we remove next. It is likely that there are leader nodes which have no loyal followers assigned to them: for such a node $v \in \mathcal{L}$, $M(v) = v$ and $M(u) \neq v$ for all $u \neq v$. For each such leader, say v, with no loyal followers, we look at all the loyal followers that neighbor it. We focus on loyal followers because they identify communities which have been seeded already. Let $u_1, \ldots, u_\ell \in \mathcal{F}$ be loyal followers such that $(u_j, v) \in E$ for $1 \leq j \leq \ell$. Find the 'community' (i.e. the leader node) to which a majority of these loyal followers are assigned (ties broken arbitrarily). Suppose v' is the leader that identifies this community, i.e. for a majority of u_j's $M(u_j) = v'$. Then set $M(v) = v'$. This step helps remove the spurious communities.

If G satisfies Definition 1, the above assignment step will result in all nodes being assigned to the proper communities, as proven in Section 5. However, in a general network, it may happen that a node only neighbors other leaders and no loyal followers. In this case, we assign the node to the community to which a majority of its neighbors belong, with ties being broken arbitrarily. We summarize the assignment step below.

Summary of community assignment step

1. Given leader set $\mathcal{L}$ and loyal follower set $\mathcal{F} = V \backslash \mathcal{L}$. Let $\mathcal{N}(v) = \{u \in V : (u, v) \in E\}$.

2. Let $\{v_1, v_2, ..., v_{|\mathcal{L}|}\}$ be ordering of nodes of $\mathcal{L}$ as per increasing distance centrality.

3. Define $M : V \to V \cup \{\star\}$ with $M(v) = v$ for all $v \in \mathcal{L}$; $M(v) = \star$ for all $v \in \mathcal{F}$.

4. For $i = 1, 2, ..., |\mathcal{L}|$, in that order, do the following: define $F_{v_i} = \{v \in \mathcal{N}(v_i) \cap \mathcal{F} : M(v) = \star\}$; for all $v \in F_{v_i}$, set $M(v) = v_i$ and $C_{v_i} = \{v_i\} \cup F_{v_i}$.

5. For each $v_i \in \mathcal{L}$ with $C_{v_i} = \{v_i\}$ do the following: delete C_{v_i}, define multiset $H_{v_i} = \{M(v) : v \in \mathcal{N}(v_i) \cap \mathcal{F}\}$:

(a) If $H_{v_i} \neq \emptyset$, let u be the maximally repeated element in H_{v_i} (ties broken arbitrarily). Then, set $M(v_i) = u$ and $C_u = C_u \cup \{v_i\}$.

(b) If $H_{v_i} = \emptyset$, define multiset $I_{v_i} = \{M(v) : v \in \mathcal{N}(v_i)\}$ and let u be the maximally repeated element in I_{v_i} (ties broke arbitrarily). Then, set $M(v_i) = u$ and $C_u = C_u \cup \{v_i\}$.

3.3 Main result

For any network satisfying Definition 1, somewhat surprisingly we find that the leader-follower algorithm that can detect the communities exactly. We note that this type of strong theoretical guarantee cannot be provided for other more common techniques such as spectral clustering. The formal statement is as follows, whose proof can be found in Section 5.

Theorem 1. *Given a network graph $G(V, E)$ with a community structure satisfying Definition 1, the leader-follower algorithm will detect all of the communities exactly with $O(|V||E|)$ operations.*

In general, a network may not satisfy Definition 1. However, the algorithm extends naturally for arbitrary networks and produces relevant communities. The experimental results presented in Section 4 confirm the effectiveness of the algorithm on real networks.

4 Empirical results

In this section we compare the performance of the leader-follower algorithm with the more common spectral clustering based on RatioCut [1]. We use the number of communities found using the leader-follower algorithm as the input for the spectral clustering algorithm. The algorithms are tested on several synthetic and real networks. In each case, we find that the leader-follower algorithm can resolve a finer community structure than spectral clustering.

4.1 Synthetic networks

We have already seen in Figure 1 that spectral clustering cannot correctly identify communities in networks satisfying Definition 1 which have dense inter-community connectivity. However, as stated in Theorem 1, the leader-follower algorithm will always correctly identify the communities in any network satisfying Definition 1, independent of the inter-community edge density. Also, it is important to note that unlike spectral clustering, the leader-follower algorithm does not need the number of communities as input. It learns this number naturally from the network structure. This is a strong advantage of the algorithm over spectral clustering.

4.2 Social networks: Facebook

The leader-follower algorithm was applied to a Facebook network. This network is the subgraph induced by the friends of a single Facebook user, i.e., the friends of the user and all edges between these friends. The network data was anonymized due to submission requirements. The results of the algorithm were striking. We found 25 different communities in the user's network. By analyzing the membership of each of these communities, we found that they corresponded to real social groups to which the user belonged, with the labels in Figure 3 indicating the unifying theme of each group. The groups corresponded to members of college cultural groups, family friends, research groups, groups of friends from a specific country, friends from a different college, and high school friends. Some groups had no specific label, but were instead groups of people who the user met through a specific person (i.e. Friends 1, Friends 2, etc.).

For comparison, the spectral clustering communities are also shown in Figure 3. As can be seen, spectral clustering ends up grouping several communities together into

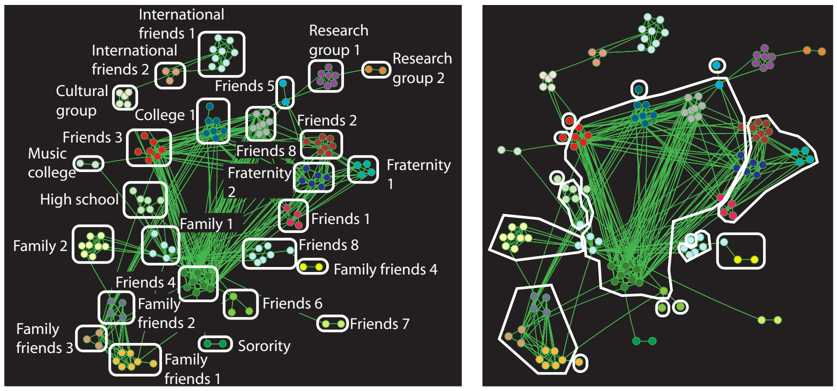

Figure 3: (left) Communities detected in a Facebook network using the leader-follower algorithm and their corresponding labels. The leader-follower communities are indicated by color and grouped together. (right) Communities detected using spectral clustering. The clusters found with spectral clustering which are not leader-follower communities are enclosed with white lines.

a large community and creates some communities consisting of single nodes. Spectral clustering is not able to resolve the finer community structure of the more densely connected communities, in contrast to the leader-follower algorithm. Also, we note that we did not need to specify beforehand the number of communities with the leader-follower algorithm. It detects this naturally from the structure of the network.

4.3 Biological networks: the human brain

We applied the algorithms to the correlation network of a human brain obtained from fMRI data [11]. Edges in the network correspond to nodes whose correlation exceeded a threshold value. Many communities were detected and much to our surprise, the leader-follower algorithm was able to identify 3 of the larger communities with major lobes of the brain which are related to different types brain functions: the frontal lobe, the occipital lobe, and the temporal lobe. Figure 4 shows the labellings of the communities (lobes) we detected using the leader-follower algorithm.

Spectral clustering was not able to resolve the temporal lobe in this network, but instead grouped it with the frontal lobe. Again, we see the leader-follower algorithm resolving finer structures than spectral clustering. This result indicates that the community structure we defined may apply to a broader class of networks beyond just social networks.

5 Proof of Theorem 1

In this section we will prove Theorem 1. Throughout we assume the network satisfies Definition 1. This proof consists of two parts. First, we show that using negative

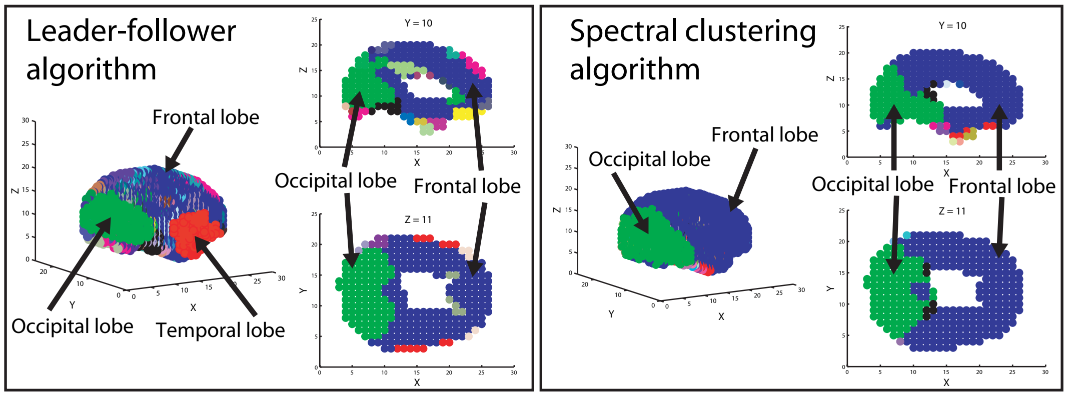

Figure 4: Lobes detected in a human brain network using (left) the leader-follower algorithm and (right) spectral clustering.

distance centrality as a score function will allow us to detect all leaders and no followers. Second, we show that the assignment step will assign the nodes to the proper communities.

5.1 Distance centrality detects leaders

Recall that we defined a follower as a node that has no neighbors outside of its community and a leader to be any node whose score was greater than at least one of its neighbors The leader-follower algorithm assigned a score to each node in the network and used this score to detect leaders. We will now show that if negative distance centrality is used as the score function, then all leaders will be detected and no followers will be detected.

Consider a network $G(V, E)$ satisfying Definition 1 with k communities of different size. For any community C_i, select any leader node $l \in C_i$ and a loyal follower $f \in C_i$. By definition, every community will have at least one leader and one follower. We will now compare the distance centralities of these two nodes.

We can divide the network G into three different sets as shown in Figure 5. First, there are nodes in community C_i. Both l and f are one hop away from all nodes in their own community. Second, there are nodes $u1$ such that $d(l, u1) = d(f, u1)$. The distances are equivalent if l is not on the shortest path from f to $u1$. In this case there is another node v in C_i that is on the shortest path to $u1$ for both l and f, and both of these nodes are one hop from v. Third, there are nodes $u2$ such that $d(f, u2) = d(l, u2) + 1$. If l is on the shortest path from f to $u2$, then because f neighbors l we have that $d(f, u2) = d(l, u2) + 1$. Also, there must be at least one node $u2$ such that this condition holds (i.e. at least one node in another community that l neighbors) otherwise l would be a follower node. We will then have for any follower f

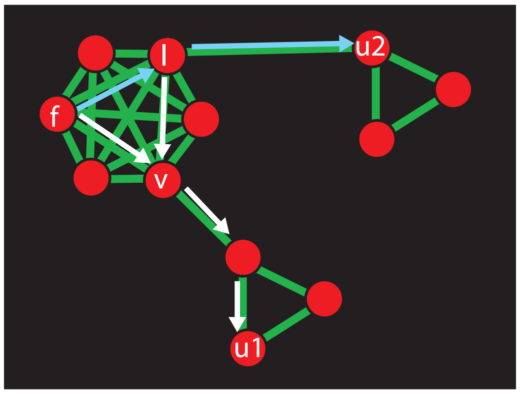

Figure 5: Node $u1$ is the same distance from the follower f and leader l because the shortest path for both nodes passes through v. Node $u2$ is closer to l than to f because l lies on the shortest path from f to $u2$. The arrows indicate the shortest path to $u1$ and to $u2$ from f and l.

and leader l in the same community

$$\begin{aligned} -\sum_{u \in G} d(f,u) &< -\sum_{u \in G} d(l,u) \\ -D(f) &< -D(l) \\ S(f) &< S(l) \end{aligned}$$

Therefore, using negative distance centrality as a score function will detect every leader in the network. Next, we must make sure that no follower is detected as a leader. Followers will not be detected as long as their score is equal all other followers in their community. To see this, note that for any followers in the same community, their distance to any node within the community is one hop, and for any node u not in the community, the distance will be $d(l,u)+1$, where l is the leader node on the shortest path from the followers to u. Thus, any followers in the same community will have the same distance centrality and will not be detected as leaders.

5.2 Community assignment

We assume that the leader detection step is done and we have a set of nodes which are leaders. The first step of the assignment process goes through the leaders in order of decreasing score and lets each leader be assigned all loyal followers which neighbor it. Since the communities are cliques and followers only have neighbors within the community, at the end of this process we will have a set of assigned nodes with cardinality greater than one for each community in the network. These sets will consist of one leader and all followers which neighbor it.

Next we are only left with leaders which received no followers. Each of these leaders neighbors at least one follower which is in their own community. The next step of the assignment process has each unassigned node join the community of a majority of its neighboring followers. Because the followers which neighbor these leaders belong to the same community as them, this step will end with all of the unassigned leaders being assigned to their proper community. Therefore, the leader-follower algorithm will have correctly detected the community structure in the network.

5.3 Computational complexity

The computation requirements of the leader-follower algorithm are dominated by the leader detection step, where one needs to calculate the distance centrality of all nodes in the network. For a network $G(V, E)$, the worst case bound on calculating distance centrality for a single node is $O(|E| + |V| \log |V|)$, which is essentially $O(|E|)$ for dense graphs. Thus, the computational complexity for calculating the distance centrality for all nodes in the network will be $O(|V||E|)$. This can be achieved using, for example, Dijkstra's algorithm [12].

Once the scores are calculated, leader detection is performed by comparing the scores of each node with its neighbors. Therefore, a comparison is made on each edge of the graph. There will be at most 2 comparisons performed on a single edge (one for each node on either side of the edge), so this step will incur $O(|E|)$ computation, which is much less than the $O(|V||E|)$ required to calculate the scores.

The assignment step involves assigning each node to a community, which cannot require more than $O(|V||E|)$ computation even for dense graphs. Therefore, the worst case computational complexity of the algorithm is $O(|V||E|)$.

6 Discussion and conclusion

We have presented a non-parametric algorithm for detecting community structure in networks known as the leader-follower algorithm. We based the algorithm upon the natural internal structure expected for a community in a social network. Because the leader-follower algorithm uses internal rather than external community properties, we expect it to be very effective in finding small scale community structure in dense networks. We proved that the algorithm is able to exactly detect communities which possess this structure. We applied the algorithm to different real networks ranging from social to biological networks, and found that it could detect relevant community structure in all of these networks. Because the leader-follower algorithm uses internal rather than external community properties, we found that it is able to detect small scale community structure that more common methods such as spectral clustering could not detect. Also, unlike spectral clustering, the leader-follower algorithm is able to learn the number of communities from the network structure and does not require it as an input.

Because of the wide range of applicability provided by the leader-follower algorithm, we would like to extend its theoretical guarantees to a larger class of communities. One strong property presented here is the requirement that the community be a clique. In reality communities are not always cliques, but rather just dense subgraphs. A question to ask is whether or not the leader-follower algorithm can provide provable guarantees when the requirement that a community be a clique is relaxed to allow subgraphs which are dense. Stated more precisely, if a community is required to not have an edge density of 1 (a clique), but rather an edge density of $(1 - \epsilon)$ for some small ϵ, then for what values of ϵ is correct community detection guaranteed? Based upon experimental results shown here, we feel that such a guarantee is possible for this type of model and that this is an interesting area for future research.

References

[1] L. Hagen and A. B. Kahng, "New spectral methods for the ratio cut partitioning and clustering", *IEEE Trans. Compututer Aided Design*, vol. 11, pp. 1074-1085, 1992.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888-905, 2000.

[3] M. B. Hastings, "Community detection as an inference problem", *Phys. Rev. E*, 74:035102, 2006.

[4] M. E. J. Newman and E. A. Leicht, "Mixture models and exploratory analysis in networks", *Proc. Natl. Acad. of Sciences*, vol. 104, pp. 9564 - 9569, 2007.

[5] J. M. Hofman and C. H. Wiggins, "Bayesian approach to network modularity", *Phys. Rev. Lett.*, 100:258701, 2008.

[6] E. M. Airoldi, and D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels", *J. Mach. Learn. Res.*, vol. 9, pp. 1981 - 2014, 2008.

[7] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", *Phys. Rev. E*, 69:026113, 2004.

[8] S. Fortunato, "Community detection in graphs", *Phys. Reports*, vol. 486, pp. 75-174, 2010.

[9] M. A. Porter, J. P. Onnela, and P. J. Mucha, "Communities in networks", *Notices of the Amer. Math. Soc.*, vol. 56, pp. 1082-1097, 2009.

[10] G. Sabidussi."The centrality index of a graph", *Psychometrika*, vol. 31, pp. 581-603, 1966.

[11] Private communication with Boon Thye Thomas Yeo and Bruce Fischl

[12] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik* vol. 1, pp. 269-271, 1959.